2017 IEEE 24th International Conference on Web Services

Mobile Application Rating Prediction via Feature-Oriented Matrix Factorization

Tingting Liang
Zhejiang University
Hangzhou, China
liangtt@zju.edu.cn

Liang Chen
Sun Yat-Sen University
Guangzhou, China
jasonclx@gmail.com

Xingde Ying
Zhejiang University
Hangzhou, China
yingxingde@zju.edu.cn

Philip S. Yu
University of Illinois at Chicago
Chicago, USA
Tsinghua University
Beijing, China
psyu@uic.edu

Jian Wu
Zhejiang University
Hangzhou, China
wujian2000@zju.edu.cn

Zibin Zheng
Sun Yat-Sen University
Guangzhou, China
zibin.gil@gmail.com

***Abstract*—With the proliferation of mobile application (app) markets (e.g., Google Play, Apple App Store), predicting user preferences on apps becomes a challenging problem. Different from previous work, we assume that a user likes an app because he/she likes certain features of the app (e.g., permission, genre, topic). Based on this assumption, we propose a feature-oriented approach to predict user preferences on apps. Specifically, we transform the original app rating matrix to feature rating data and predict the unknown ratings on the features through a latent factor model, instead of directly predicting ratings on apps. The predicted user ratings on features can be used to generate the ratings on apps. Two integration methods are presented to give different significance for feature preferences. The approach has some obvious advantages: as it integrates feature information to analyze the details of user preference, it can generalize better as the feature rating data is denser, and improve the interpretation of the prediction of app ratings. Experimental results on a real-world dataset demonstrate the effectiveness of the proposed approach.**

***Keywords*-rating prediction; latent factor model; feature-oriented prediction; app recommendation;**

I. Introduction

With the explosive growth of mobile apps in the existing markets, e.g., Google Play[1], Apple App Store[2], it becomes more difficult for users to discover the appropriate ones. Recommender systems for apps can relax this situation to a great extent, whose objective is to predict user preferences on apps through the historical preferences. Costa-Montenegro et al. [1] designed an integrated solution that makes automatic app recommendation for users through monitoring users' activity. Yin et al. [2] presented a model to capture the contest between apps and integrate it into app recommendation. Lin et al. [3] applied a semi-supervised topic model to characterize version features into a set of latent topics, and discriminate the topics based on genre information to obtain a recommendation score for an app's version for a target user. The basic idea of these methods is predicting user preferences on apps based on the preferences inferred from the downloaded ones.

Different from the conventional approaches that concentrated on either apps or users, this work tries to consider the app recommendation problem from another side, focusing on the features of apps (feature-oriented). Each app has many characteristics (e.g., permissions, topics, genres) due to its multiple functions. For example, one of the permissions Google Map app has is getting users' precise location as it can offer navigation service. And it also has the access to microphone so that it can get users' requirements through voice. If a user downloads an app, he/she may be attracted by several features of the app rather than all the features. That is, two users may like the same app because of different reasons. A user may like the theme of a game app but not its interface, while another user may accept the permission of getting location but not the permission of reading call log or something else. If this is the case, the traditional app-oriented (user-oriented) approaches may not work.

We illustrate the difference between traditional methods and feature-oriented method based on a simple example with four apps as shown in Fig. 1. The solid arrow indicates the user likes the app while the dashed arrow means the app is recommended to the user. Suppose that user *A* likes *Plants vs. Zombies* and *Plants Attack* due to feature "Adventure", and user *B* loves *Plants vs. Zombies*, *Plants Attack*, and *Plantera* due to the character feature "Plant". Traditional collaborative filtering will recommend *Plantera* to user *A* (the red dashed arrow) as *A* and *B* both love *Plants vs. Zombies* and *Plants Attack*. However, user *A* would prefer *Clash of Clans* since he pays more attention on the feature "Adventure".

Motivated by the above discussion, we will be able to predict a user's interest in an app if we can figure out

[1]Google Play: https://play.google.com/store/apps
[2]Apple App Store: https://itunes.apple.com/us/genre/ios/id36?mt=8

978-1-5386-0752-7/17 $31.00 © 2017 IEEE
DOI 10.1109/ICWS.2017.41

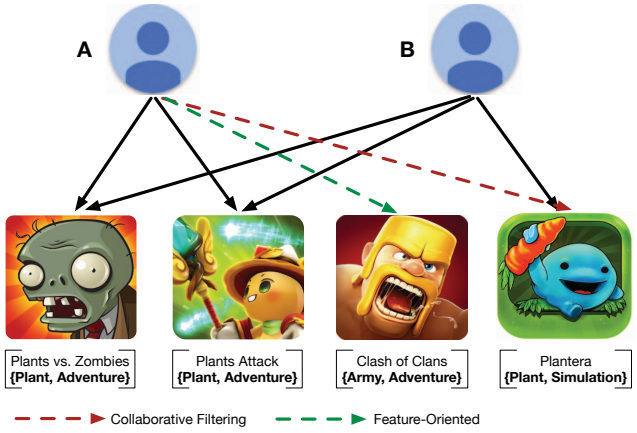


Figure 1. Example of Collaborative Filtering and Feature-Oriented Recommendation

the preference of the user on features as the features are valid representations of apps. Therefore, if users' observed preferences on features can be expressed in the same form as users' observed preferences on apps, most of the conventional app rating prediction algorithms can be employed to predict user preferences on apps. Following this idea, this paper proposes a feature-oriented matrix factorization method called FOMF to predict user preferences on apps. Specifically, given the user-app rating records, we first convert app ratings into feature ratings and obtain a user-feature rating matrix. Latent factor models are applied to the generated user-feature rating matrix to predict the unknown user preferences on app features. To predict a user's rating on an unrated app, we need to combine the predicted ratings of related features to obtain the rating of the app. Two integration strategies are presented, one is based on average, and the other is based on regression. Average based strategy treats all the features equally while regression based strategy gives different significance for feature preferences.

In particular, the main contributions of our paper are summarized as follows:

- We propose a feature-oriented matrix factorization called FOMF for app rating prediction. FOMF first predicts unrated feature ratings based on the ratings converted from user-app rating records, and then integrates feature ratings to derive user ratings on apps.
- We present two integration strategies that are respectively based on average and regression, to effectively integrate feature ratings.
- Empirical studies based on a real world dataset crawled from Google Play demonstrate the effectiveness of the proposed FOMF.

The remainder of paper is structured as follows. Section II discusses related work. The proposed feature-oriented matrix factorization method is described in Section III. Section IV presents the concrete evaluation process and performance analysis. Finally we make a conclusion in Section V.

II. Related Work

A. Recommendation for Mobile Applications

To alleviate information overload in app market, plenty of app recommendation methods have been proposed. Some of these works focused on employing user information (e.g., usage records, personal preferences) to improve recommendation quality . Yan et al. [4] built an app recommendation system to make personalized application recommendations by analyzing the usage records of users' installed applications. In [5], Liu et al. proposed to incorporate both interest-functionality interactions and user preferences on privacy to produce personalized app recommendations. Zheng et al. [6] developed a user-centered method to mine knowledge from GPS trajectory data to make app recommendations on locations and activities. Yang et al. [7] leveraged apps as features to describe users' personal interests and designed an approach to produce recommendation for the target user.

There exist some other works that prefers to use features of apps (e.g., versions, social information) to improve the performance of app recommendation. Xu et al. [8] investigated the diverse usage patterns of smartphone apps through network measurements from a national level tier-1 cellular network provider in the U.S. In [9], Lin et al. presented an approach that accounts for nascent information extracted from apps' Twitter accounts to provide recommendation for apps in cold-start situation. Kim et al. [10] defined semantic relations between apps consumed by a specific member and his social members, and proposed a recommendation system for mobile users based on this semantic relations.

The remaining works considered both user and app information. Yin et al. [2] considered behavioral factors that invoke a user to replace an old app with a new one and presented that each owned app has an actual satisfactory value and a new app under consideration has a tempting value, based on which they built a model to capture the contest between apps and integrate it into app recommendation. Zhu et al. [11] developed techniques to automatically detect the potential security risk for each app by exploiting the requested permissions, and then proposed a flexible approach for recommendation based on modern portfolio theory considering both apps' popularity and users' security preferences.

All of the mentioned works concentrate on either apps or users. In this paper, the proposed feature-oriented matrix factorization method captures the user preferences on features by performing latent factor models on user-feature ratings. The prediction of app ratings are derived through the integration of the predicted feature ratings.

B. Latent Factor Models

Recently, latent factor models has shown its effectiveness in recommender systems, which factorize user-item rating

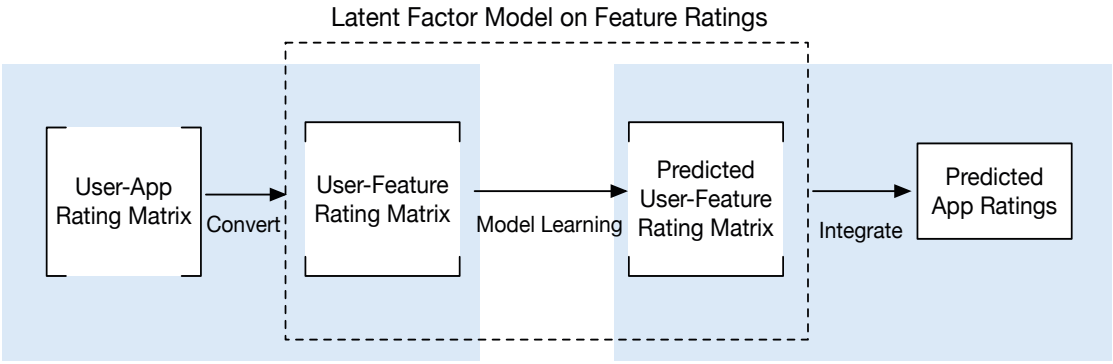


Figure 2. Framework of Feature-Oriented Matrix Factorization (FOMF) for App

matrix into two low rank user-specific and item-specific matrices. Simon Funk firstly posed a revised SVD method on his website[3], which attracted much attention to the latent factor models. Since than, many SVD-based models were proposed, e.g., Bias-SVD, SVD++ [12]. Salakhutdinov and Mnih proposed probability distribution on the latent feature vectors and developed probabilistic matrix factorization (PMF) [13][14]. Based on the idea of PMF, many variants and extensions have been proposed. The regression-based latent factor model [15] incorporated features and past interactions to regress the latent vectors. The collaborative topic regression (CTR) [16] studies the recommendation for articles with each article being modeled by topic modeling on the text content.

III. Feature-Oriented Matrix Factorization

A. Overview

This paper proposes a feature-oriented matrix factorization method to employ user preferences on features to improve the accuracy of app rating prediction. As shown in Fig. 2, FOMF consists of three main steps. The first step is to extract user ratings on features from user-app rating matrix and construct a user-feature rating matrix. The second step is the part of the dashed box in Fig. 2, and we consider latent factor model for this box to predict user preferences on app features. The last step is integrating feature ratings to derive user ratings on apps.

B. User-Feature Rating Matrix Construction

Suppose there are I user, J apps with T features. The original user-app rating records can be denoted by an $I \times J$ matrix, where each entry r_{ij} represents the preference of user i on app j. Each app j is associated with a set of features denoted by F_j. For a feature t, there exists a set of apps j denoted by A_t such that $t \in F_j$.

We take $z_{it}(j)$ as the rating given by user i on feature t once user i rates an app j associated with feature t. For user i and feature t, $\{z_{it}(j)\}$ denotes the set of ratings $z_{it}(j)$ for all apps $j \in A_t$. Based on the user-feature rating sets, we construct an $I \times T$ user-feature rating matrix denoted by Z. Each row of Z indicates a user i, each column indicates a feature t, and the entry for (i, t) is the rating set $\{z_{it}(j)\}$. Note that some rating sets for pairs (i, t) are very large which may reduce the efficiency. There are several ways to deal with this situation and we randomly select 10 ratings from those large rating sets in this paper.

The proposed model FOMF uses two kinds of features, app permissions and topics. For app permissions, we consider that the user preferences on permissions are different since they are directly related to the user privacy. For app topics, each app may have multiple topics due to its multiple functions, which can be mined from its text descriptions. We adopt the widely used Latent Dirichlet Allocation (LDA) model [17] to extract app topics as features. The effectiveness of these two features will be discussed in the evaluation part.

C. Latent Factor Model on Feature Ratings

We adopt Bias-SVD model [18] on user-feature matrix Z to generate the latent user vector p_i for each user i and the latent feature vector q_t for each feature t. Bias-SVD model adds bias terms β_i and β_t into the simple latent factor model. The simple latent factor model tries to capture the interactions between users and features that generate the different rating values. However, the rating values are influenced not only by the interactions but also by biases associated with either user or features. The bias terms β_i and β_t respectively denote the observed deviations of user i and feature t from the average. In this model, a user-feature rating is predicted by the rule:

$$\hat{z}_{it} = \mu + \beta_i + \beta_t + p_i^T q_t \quad (1)$$

where μ is the overall average rating. Here, the rating is broken down into its four components: global average, user bias, feature bias, and user-feature interaction.

[3]Simon Funk website: http://sifter.org/ simon/journal/20061211.html

To learn the model parameters, we should minimize the regularized squared error function:

$$min \sum_{(i,t)\in B} (z_{it} - \hat{z}_{it})^2 + \lambda(\beta_i^2 + \beta_t^2 + ||p_i||^2 + ||q_t||^2) \quad (2)$$

Here, B is the set of the (i,t) pairs for which z_{it} is known (the training set). The goal of the model is to generalize the previous ratings in a way that predicts future ratings. Thus, it is important to avoid overfitting problem through regularizing the learned parameters. The constant λ is the parameter that controls the extent of regularization. Salakhutdinov and Minh [14] provides a probabilistic foundation for regularization.

Stochastic gradient descent method is used to get the update rule for each parameter:

$$\begin{aligned} \beta_i &\leftarrow \beta_i + \eta(e_{it} - \lambda\beta_i) \\ \beta_t &\leftarrow \beta_t + \eta(e_{it} - \lambda\beta_t) \\ p_i &\leftarrow p_i + \eta(e_{it}q_t - \lambda p_i) \\ q_t &\leftarrow q_t + \eta(e_{it}p_i - \lambda q_t) \end{aligned} \quad (3)$$

where $e_{it} = z_{it} - \hat{z}_{it}$ and η is the learning rate.

D. Integration Strategy

Based on the model discussed above, user i's predicated rating on feature t is denoted as $\hat{z}_{it} = \mu + \beta_i + \beta_t + p_i^T q_t$. The problem now is how to integrate user-feature ratings to derive accurate ratings on apps. In this paper, we present average based strategy and regression based strategy for predicted feature ratings integration.

1) Average-based: This strategy uses the average of feature ratings to predict the user ratings $\hat{r}_{ij}$ for app j. The prediction formula is defined as:

$$\hat{r}_{ij} = \frac{1}{|F_j|} \sum_{t\in F_j} \hat{z}_{it} \quad (4)$$

This integration strategy treats all features in F_i equally as each $\hat{z}_{it}$ has the same weight $1/|F_j|$. It does not consider the user-specific information, but only the feature information of apps.

2) Regression-based: In addition to the average based strategy, we present another integration method which employs support vector regression (SVR) model [19] to automatically learn the weighting for features.

In this strategy, we assume that there is a $T-$dimension weighting vector $\mathbf{w}$ and each element w_t indicates the importance of each feature t. Based on the training data, suppose $\mathbf{x}_k$ is the kth input data and y_k is the kth output data. Specifically, we set $y_k = r_{ij}$ if the rating is made by user i on app j, and set $\mathbf{x}_k(t) = \mu + \beta_i + \beta_t + p_i^T q_t$ if $t \in F_j$ and $\mathbf{x}_k(t) = 0$ otherwise. The goal of SVR is to find a function $y_k = \mathbf{w}^T\mathbf{x}_k + b_i$ that has at most ε deviation from the actually obtained targets y_k for all the training data. In order to find the optimal weighting vector $\mathbf{w}$ and user specific bias b_i, the following optimization problem should be solved:

$$\begin{aligned} min &\frac{1}{2}||\mathbf{w}||^2 + \frac{C}{2}\sum_k(\xi_k^2 + \xi_k^{*2}) \\ s.t. \quad &|\mathbf{w}^T\mathbf{x}_k + b_i - y_k| \le \varepsilon + \xi_k; \\ &\xi_k, \xi_k^* \ge 0; \forall k. \end{aligned} \quad (5)$$

where ξ_k, ξ_k^* are two slack variables and C is a constant.

The Lagrangian is,

$$\begin{aligned} L =& \frac{1}{2}||\mathbf{w}||^2 + \sum_k \alpha_k(\mathbf{w}^T\mathbf{x}_k + b_i - y_k - \varepsilon - \xi_k) \\ &+ \sum_k \alpha_k^*(y_k - \mathbf{w}^T\mathbf{x}_k - b_i - \varepsilon - \xi_k^*) \\ &+ \frac{C}{2}\sum_k(\xi_k^2 + \xi_k^{*2}) \end{aligned} \quad (6)$$

where α_k and α_k^* are Lagrange multipliers. Take the derivatives with respect to $\mathbf{w}$, b_i, ξ_k, ξ_k^*, leading to the KKT conditions [20] as:

$$\mathbf{w} = \sum_i(\alpha_k - \alpha_k^*)x_k, \quad \xi_k = \alpha_k/C, \quad \xi_k^* = \alpha_k^*/C \quad (7)$$

The comprehensive inference will not be presented in this paper. More details can be found in [19]. After the optimal weighting vector $\mathbf{w}$ and the bias term b_i are determined, the predicted rating of user i on app j can be generated by:

$$\hat{r}_{ij} = \sum_{t\in F_j} w_t\hat{z}_{it} + b_i \quad (8)$$

IV. Experiments

In this section, we will verify the effectiveness of the proposed feature-oriented matrix factorization method by conducting a series experiments compared to several well known baselines.

A. Dataset Description

We employ a real-world dataset crawled from the most popular Android app market, Google Play. As Fig. 3 shows, each app in Google Play has its own description page, from which we collect app's meta data, including its name, genre, permissions, average user rating score, description, etc. In addition to the meta data, the dataset includes user reviews of each app. Each user review consists of a user rating, the related comment in the text form, and a timestamp showing when the review was created.

We remove apps with few reviews and users who rarely rated apps to avoid cold start problem. Specifically, we first remove apps with less than 25 reviews and then remove users with less than 4 apps. After the preprocessing step, we get a dataset consisting of 7,536 users, 11,390 apps and 59,608 rating records for evaluation. The user-app rating matrix has a sparsity as high as 99.93%. For the app features, the dataset

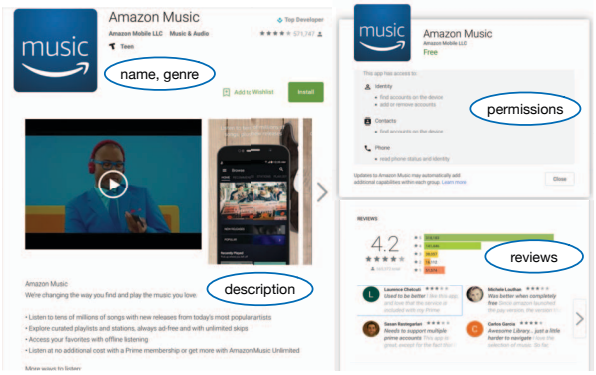


Figure 3. App Page in Google Play

Table I
DATA DESCRIPTION

Users	Apps	Ratings	Sparsity	Permissions
7,536	11,390	59,608	99.93%	84

includes 84 different permissions and the number of topics is to be decided. In the comparison evaluation part, we set the number of topics as 50. Table I shows basic statistics of the employed app dataset.

B. Evaluation Metrics

We use Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) to evaluate the performance of the proposed model and other baselines. The definition of MAE is:

$$MAE = \frac{1}{N}\sum_{i,j}|R_{ij} - \hat{R}_{ij}| \tag{9}$$

where R_{ij} represents the rating user i gave to app j, and $\hat{R}_{ij}$ denotes the rating predicted by a method. Moreover, N is the number of tested ratings. The metric RMSE is defined as:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i,j}(R_{ij} - \hat{R}_{ij})^2} \tag{10}$$

A smaller MAE or RMSE means better performance.

C. Comparison

To prove the superiority of the proposed feature-oriented matrix factorization method, we compare it with three baselines which are summarized as follows:

- **PMF**: This method is a typical matrix factorization method proposed by Salakhutdinov and Minh [13].
- **Bias-SVD**: This model adds bias of user and item into the simple latent factor model [18]. And it is used for the user-feature matrix factorization in this paper.
- **WMF**: This method [21] captures user-specific interests to predict ratings on apps.

There are different variations for our approach FOMF described as follows:

- **FOMF-P**$_a$: This variation uses app permission as the feature and employs average integration strategy.
- **FOMF-P**$_r$: This variation uses app permission as the feature and employs regression integration strategy.
- **FOMF-T**$_a$: This variation uses app topic information as the feature and employs average integration strategy. The topic information is extracted from app descriptions by LDA model.
- **FOMF-T**$_r$: This variation uses app topic information as the feature and employs regression integration strategy. The topic information is extracted from app descriptions by LDA model.
- **FOMF-M**$_a$: This variation uses the features combining app permission and topic information and employs average integration strategy.
- **FOMF-M**$_r$: This variation uses the features combining app permission and topic information and employs regression integration strategy.

Next we will validate the effectiveness of FOMF through the comparison between its different variations and baselines. We adopt the dimensionality of $K = 15$ for latent vectors and the learning rate of $\eta = 0.01$ for all the methods. For FOMF-P$_a$, FOMF-T$_a$, FOMF-M$_a$, and FOMF-M$_r$, the value of λ is set to 0.1. And for FOMF-P$_r$ and FOMF-T$_r$, the value of λ is set to 0.01. The parameters of baselines are set to the optimal values. We arrange the app rating records in chronological order and use different ratios (40%-80%) of these records as training data. For example, the training data 70% means that we select the first 70% of the ordered ratings as the training data to predict the remaining 30% of ratings.

Table II shows the results of different methods. From the experimental comparisons, we can make the following observations.

- The proposed FOMF always outperforms the baselines in most conditions. The lowest values of MAE and RMSE are mainly generated by FOMF-M$_r$, FOMF-T$_a$. It validates the power of the proposed feature-oriented matrix factorization method that takes app features as the center in the prediction process by converting the original user-app rating matrix into user-feature rating matrix.
- Among the two integration strategies, the average based strategy (i.e., FOMF-P$_a$, FOMF-T$_a$, and FOMF-M$_a$) performs better on the metric RMSE while regression based strategy (i.e., FOMF-P$_r$, FOMF-T$_r$, and FOMF-M$_r$) gets a lower MAE value. It illustrates that the presented strategies have different strengths on the prediction performance. For the same integration, adopting app topic as the feature leads to a better performance compared with app permission information. It implies

Table II
PERFORMANCE COMPARISON (THE BASELINE OF IMPROVED PERFORMANCE IS PMF)

Training	Metrics	PMF	Bias-SVD	WMF	FOMF-P_a	FOMF-T_a	FOMF-M_a	FOMF-P_r	FOMF-T_r	FOMF-M_r
40%	MAE	1.4127	1.0681	1.0098	1.0374	1.0052	1.0058	1.0096	1.0038	**1.0029**
	Improve		24.39%	28.52%	26.57%	28.84%	28.80%	28.54%	28.94%	29.01%
	RMSE	1.8776	1.6511	1.6400	1.5174	**1.4959**	1.5064	1.5630	1.5356	1.5302
	Improve		12.07%	12.66%	19.19%	20.33%	19.77%	16.76%	18.22%	18.50%
50%	MAE	1.3834	1.0938	1.0122	1.0321	1.0049	1.0051	1.0090	0.9986	**0.9962**
	Improve		20.93%	26.83%	25.39%	27.36%	27.34%	27.06%	27.81%	27.99%
	RMSE	1.8222	1.6357	1.6323	1.5049	**1.4873**	1.4931	1.5617	1.5119	1.5091
	Improve		10.24%	10.42%	17.42%	18.38%	18.07%	14.30%	17.03%	17.19%
60%	MAE	1.3894	1.1131	1.0246	1.0400	1.0110	1.0103	1.0144	1.0000	**0.9987**
	Improve		19.89%	26.26%	25.15%	27.24%	27.28%	26.99%	28.03%	28.12%
	RMSE	1.8224	1.6440	1.6411	1.5044	**1.4882**	1.4939	1.5776	1.5150	1.5155
	Improve		9.79%	9.94%	17.45%	18.34%	18.03%	13.43%	16.87%	16.84%
70%	MAE	1.3648	1.1330	1.0330	1.0243	1.0173	1.0098	1.0206	1.0006	**0.9935**
	Improve		16.98%	24.31%	24.95%	25.46%	26.01%	25.22%	26.69%	27.21%
	RMSE	1.7700	1.6160	1.6345	1.4858	**1.4686**	1.4773	1.5782	1.4982	1.5075
	Improve		8.70%	7.66%	16.06%	17.03%	16.54%	10.84%	15.36%	14.83%
80%	MAE	1.3587	1.1415	1.0428	1.0359	1.0331	1.0312	1.0333	1.0098	**1.0077**
	Improve		15.99%	23.25%	23.76%	23.97%	24.11%	23.95%	25.68%	25.83%
	RMSE	1.7634	1.6252	1.6423	1.4931	1.4868	**1.4867**	1.5987	1.5221	1.5245
	Improve		7.84%	6.87%	15.33%	15.69%	15.69%	9.34%	13.69%	13.55%

that topic information can better reflect user preferences on apps. The variation using the features combining app permission and topic obtains the best performance overall. One possible reason for some slight worse performance is that the sparsity of the user-feature data increases as the number of features increases.

- For the regression based approach, FOMF-M_r considering both types of features performs the best, and for the average based approach, FOMF-T_a generally does better, though FOMF-M_a is a close second.
- WMF performs better than the other two baselines which proves that incorporating user-specific interests is helpful. However, it is also beaten by our FOMF method, especially when the topic information of app is utilized. It is mainly because WMF focus on calculating the weights of each app for target users and introducing them into matrix factorization to predict app ratings, which doesn't further consider the user preferences on app features.
- When compare FOMF with Bias-SVD, we can find that conducting the same latent factor model on different rating matrix leads to quite different results. The proposed FOMF performs better due to the incorporation of feature information and the denser user-feature rating data.
- When consider different training data ratios, it can be found that the superiority of FOMF is more significant for less training data. It demonstrates that FOMF can alleviate data sparsity problem.

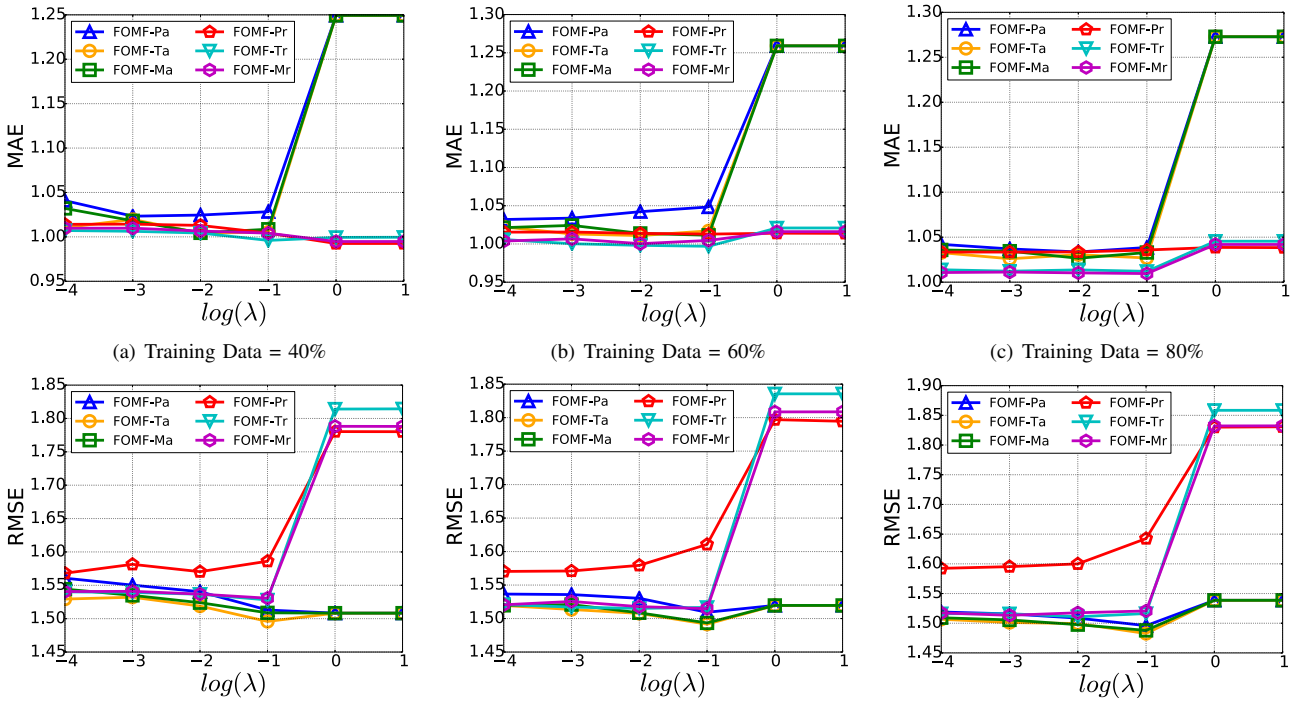
D. Impact of λ

In this section, we will study the impact of the regularization parameter λ in our model. We set dimensionality $K = 15$ and learning rate $\eta = 0.01$ for training data ratio 40%, 60%, and 80%.

The results generated by different λ are shown in Fig. 4. As analyzed above, the methods utilizing average based integration strategy (i.e., FOMF-P_a, FOMF-T_a, and FOMF-M_a) performs better on RMSE, while the methods with the other integration strategy leads to a better performance on MAE. The same phenomena happens in the study of parameter λ. It is obvious that $\lambda = 0.1$ is a pivotal point. MAE values of FOMF-M_a, FOMF-T_a, and FOMF-P_a increase rapidly after $\lambda = 0.1$, and MAE curves of FOMF-M_r, FOMF-T_r, and FOMF-P_r are relatively stable. The change of RMSE is contrary to MAE. Overall, when the regularization parameter λ is set to 0.1 or 0.01, the proposed FOMF model can generate optimal results in prediction accuracy.

E. Impact of the number of topics

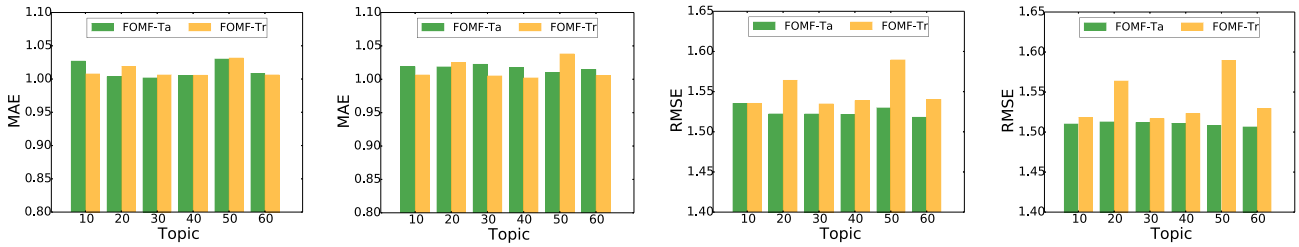
Among the variations of FOMF, the methods using app topic information can perform better in prediction accuracy. In the process of topic extraction, the number of topics is an important parameter. To study the impact of the number of



(a) Training Data = 40% (b) Training Data = 60% (c) Training Data = 80%

(d) Training Data = 40% (e) Training Data = 60% (f) Training Data = 80%

Figure 4. Impact of λ on MAE and RMSE



(a) Training Data = 40% (b) Training Data = 60% (c) Training Data = 40% (d) Training Data = 60%

Figure 5. Impact of Topic on MAE and RMSE

topics on FOMF-T$_a$ and FOMF-T$_r$, we fix other parameters and consider two different ratios of training data.

Figure 5 shows the performance of the proposed FOMF models based on different topics. It can be found that both variations perform better when the number of topics is set to 30 or 40, which means the number of topics can not be too large or too small. Compared to FOMF-T$_a$, FOMF-T$_r$ is more sensitive to the number of topics as the metric values fluctuate more obviously when the number of topics changes. Under different ratios of training data, the fluctuation of metric values with the increase of the number of topics is similar.

V. Conclusion

The previous work about app recommendation mainly focus on users or apps, and features of apps are generally considered as additional information to improve the quality of user preferences prediction. In this paper, we propose a feature-oriented matrix factorization method called FOMF to predict user ratings on apps, which is motivated by a simple observation: a user likes an app because he likes several features of the app rather than all the features. Our contribution is to transform the user-app rating prediction problem to user-feature rating prediction problem, and utilize the prediction result to derive user ratings on apps through two effective integration strategies. One advantage of FOMF is that it incorporates feature information to analyze the details of user preference, which could lead to a better prediction performance. Another benefit is that FOMF performs a latent factor model on the denser feature ratings which could relax the sparsity problem of rating data.

ACKNOWLEDGMENT

This work is supported in part by the Natural Science Foundation of China under grant of No. 61379119, No. 61672453, and No. 61672313, NSF through grants IIS-1526499, and CNS-1626432.

REFERENCES

[1] E. Costa-Montenegro, A. B. Barragáns-Martínez, and M. Rey-López, "Which app? a recommender system of applications in markets: Implementation of the service for monitoring users interaction," *Expert systems with applications*, vol. 39, no. 10, pp. 9367–9375, 2012.

[2] P. Yin, P. Luo, W.-C. Lee, and M. Wang, "App recommendation: a contest between satisfaction and temptation," in *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM, 2013, pp. 395–404.

[3] J. Lin, K. Sugiyama, M.-Y. Kan, and T.-S. Chua, "New and improved: modeling versions to improve app recommendation," in *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. ACM, 2014, pp. 647–656.

[4] B. Yan and G. Chen, "Appjoy: personalized mobile application discovery," in *Proceedings of the 9th international conference on Mobile systems, applications, and services*. ACM, 2011, pp. 113–126.

[5] B. Liu, D. Kong, L. Cen, N. Z. Gong, H. Jin, and H. Xiong, "Personalized mobile app recommendation: Reconciling app functionality and user privacy preference," in *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. ACM, 2015, pp. 315–324.

[6] V. W. Zheng, B. Cao, Y. Zheng, X. Xie, and Q. Yang, "Collaborative filtering meets mobile recommendation: A user-centered approach." in *AAAI*, vol. 10, 2010, pp. 236–241.

[7] C. Yang, T. Wang, G. Yin, H. Wang, M. Wu, and M. Xiao, "Personalized mobile application discovery," in *Proceedings of the 1st International Workshop on Crowd-based Software Development Methods and Technologies*. ACM, 2014, pp. 49–54.

[8] Q. Xu, J. Erman, A. Gerber, Z. Mao, J. Pang, and S. Venkataraman, "Identifying diverse usage behaviors of smartphone apps," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. ACM, 2011, pp. 329–344.

[9] J. Lin, K. Sugiyama, M.-Y. Kan, and T.-S. Chua, "Addressing cold-start in app recommendation: latent user models constructed from twitter followers," in *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2013, pp. 283–292.

[10] J. Kim, S. Kang, Y. Lim, and H.-M. Kim, "Recommendation algorithm of the app store by using semantic relations between apps," *The Journal of Supercomputing*, vol. 65, no. 1, pp. 16–26, 2013.

[11] H. Zhu, H. Xiong, Y. Ge, and E. Chen, "Mobile app recommendations with security and privacy awareness," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 951–960.

[12] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 426–434.

[13] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *Advances in Neural Information Processing Systems*, 2008, pp. 1257–1264.

[14] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using markov chain monte carlo," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 880–887.

[15] D. Agarwal and B.-C. Chen, "Regression-based latent factor models," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 19–28.

[16] C. Wang and D. M. Blei, "Collaborative topic modeling for recommending scientific articles," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 448–456.

[17] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[18] Y. Koren, R. Bell, C. Volinsky *et al.*, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[19] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

[20] K. Kuhn, "H. and tucker, aw, nonlinear programming," in *Second Berkeley Symposium of Mathematical Statistics and Probability*, 1951.

[21] J. Meng, Z. Zheng, G. Tao, and X. Liu, "User-specific rating prediction for mobile applications via weight-based matrix factorization," in *2016 IEEE International Conference on Web Services (ICWS)*. IEEE, 2016, pp. 728–731.